Including Hardware/Software Co-design in the ASSERT Model Driven Engineering Process *

Francisco Ferrero[1], Elena Alaña[1], Ana Isabel Rodríguez[1], Juan Zamorano[2], and Juan A. de la Puente[2]

[1] GMV, E-28760 Madrid, Spain,
fferrero@gmv.com,ealana@gmv.com,airodriguez@gmv.com
[2] Universidad Politécnica de Madrid (UPM), E-28040 Madrid, Spain,
jzamora@fi.upm.es,jpuente@dit.upm.es

Abstract. The ASSERT project defined new software engineering methods and tools for the development of critical embedded real-time systems in the space domain. The ASSERT model-driven engineering process was one of the achievements of the project and is based on the concept of property-preserving model transformations. The key element of this process is that non-functional properties of the software system must be preserved during model transformations. Properties preservation is carried out through model transformations compliant with the Ravenscar Profile and provides a formal basis to the process. In this way, the so-called Ravenscar Computational Model is central to the whole ASSERT process. This paper describes the work done in the HWSWCO study, whose main objective has been to address the integration of the Hardware/Software co-design phase in the ASSERT process. In order to do that, non-functional properties of the software system must also be preserved during hardware synthesis.
Keywords: Ada 2005, Ravenscar profile, Hardware/Software co-design, real-time systems, high-integrity systems, ORK.

1 Overview

Embedded systems are commonly built by specifying and developing independently the hardware and software subsystems from the system specification phase and their integration is performed lately at the end of the system development. However, current embedded systems are more complex and have shorter time-to-market and thus it is needed to adopt new techniques to reduce time and costs by simplifying the hardware/software integration process. The hardware/software co-design exploits the trade-offs between hardware and software in a system by developing concurrently both systems. The HW/SW co-development approach mainly consists of specifying the system functions (typically in a behavioural form) in a representation that is independent from the underlying execution platform, modelling the system platform describing the hardware architecture of the target platform, partitioning the system into either hardware or software, implementing the hardware and software systems, and scheduling the execution of the tasks to meet any timing constraints.

The ASSERT Project[3] was aimed at defining new software engineering methods and tools for the development of critical embedded real-time systems in the space domain. One of its main achievements is a new model-driven software process, which is based on the concept of property-preserving model transformations. The key element of the ASSERT process is that non-functional properties must be preserved during all phases of model transformations. Therefore, the different views of models

* This work has been funded by ESA, contract no. 22810/09/NL/JK (HWSWCO).
[3] Automated proof-based system and software engineering for real-time systems, see www.assert-project.net.

must be consistent and thus a a common meta-model was defined to provide formal basis to the ASSERT process. Moreover, the resulting software architecture must be amenable for response time analysis and thus that meta-mode was derived from the Ada Ravenscar Profile [1] and is called the Ravenscar Computational Model. The required real-time behaviour must be guaranteed at execution time and thus the ASSERT execution platform or Virtual Machine (VM) is bound to a GNAT cross-compilation system that only accepts Ravenscar Profile compliant code. In order to ensure compatibility with the Ravenscar Profile, application-level software is built in such a way that all functional code and data is encapsulated into *VM-Level Containers*, (VMLC) which are the run-time code entities that operate on top of the VM. Accordingly, VMLC are coded in Ada 2005 [2] restricted by the Ravenscar profile. However, the functional code can be written in Ada, C or C++.

This paper provides the feedback on the research activities carried out in the Hardware-Software Co-design discipline and the conclusions extracted from HWSWCO ESA study. The study has investigated the HW/SW co-design phase to integrate this engineering task as part of the ASSERT process and make it compatible with its approach, process and the existing tool-set called TASTE[4].

2 The HW/SW Co-development Methodology

The HW/SW co-development methodology is commonly divided into four different phases: co-specification, co-design, co-synthesis and co-validation. The proposed methodology starts from an abstract system description based on system behaviour and generates the architecture gradually adding implementation details to the design. In the context of HWSWCO study, the first three phases were investigated to ensure consistency between hardware and software design and verification. Figure 1 shows the proposed methodology that is based on an iterative process compliant with the ASSERT process.

Co-specification: the system functionality is described independently of the system architecture and constitutes a unified and unbiased representation of the system. The system model encloses the System Logic View and System Platform View following the same model-driven approach proposed in ASSERT.

Co-design: the system components are mapped to processing resources and the feasibility of the partition is then evaluated (which system functions are mapped either to software or hardware). The feasibility analysis is accomplished by performing high-level estimation analysis which provides figures about the use of the platform resources. Those figures will drive the partitioning process although it is necessary calculating the Worst Case Execution Time of the sequential code of system components of the software system and then a schedulability analysis in order to assess partition scheme.

Co-synthesis: the HW and SW systems are synthesized and integrated. In order to do this integration, communication interfaces between the HW and SW systems have to be generated. The usage of a common data representation and a unified system model makes it easier and faster enabling the automatic synthesis of the communication interfaces for communicating both systems.

The implementation of Co-specification, Co-design and Co-synthesis phases for this study is supported by a set of standard notations and tools that are included in Figure 1 together with the output of each phase.

[4] The ASSERT Set of Tools for Engineering

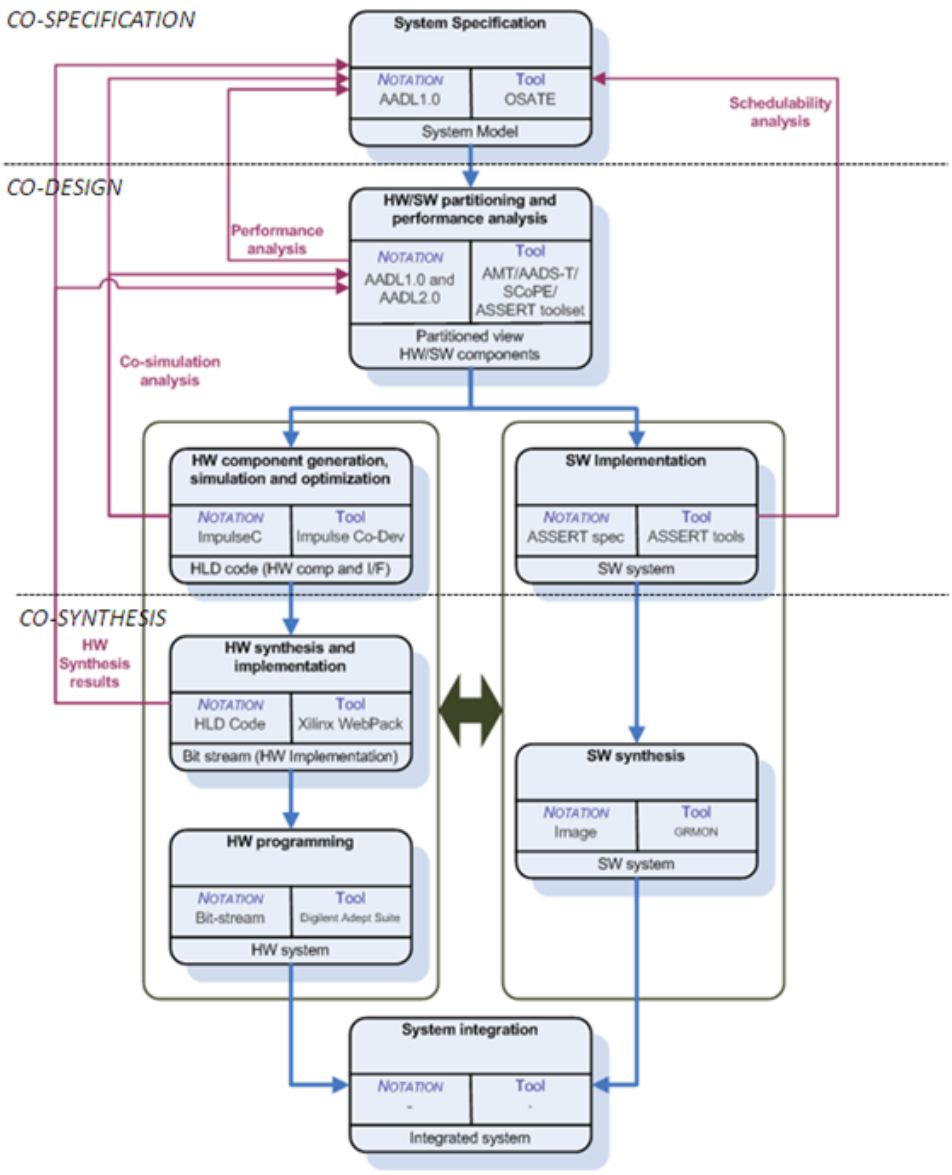

Fig. 1. HWSWCO methodology, notations and tools.

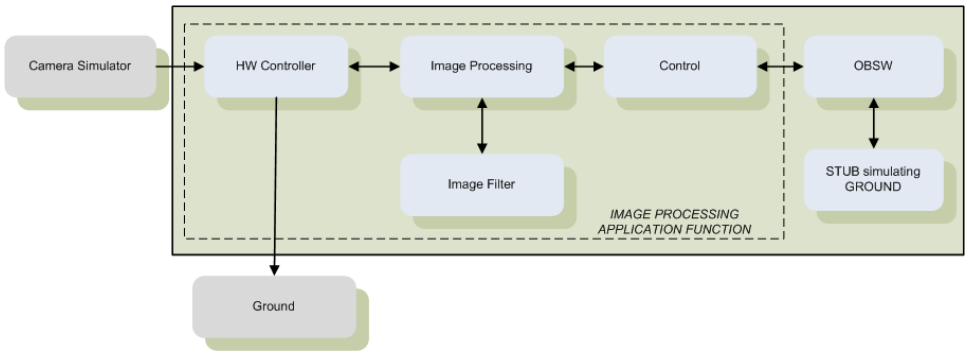

Fig. 2. Image processing case-study.

3 ASSERT Model Transformation

GMV's ASSERT Model Transformation (AMT) tool supports different Model-to-Model transformations to be performed during the co-design phase. It has been developed as an Eclipse plug-in integrated with the OSATE editor. The performance analysis performed during the co-design phase is supported by AADS-T and SCoPE tools, developed by the University of Cantabria. These tools provide the following features:

- Implementing the generation of SystemC in SCoPE compliant with the Ravenscar Computational Model.
- Implementing the estimation of the bus load in SCoPE.
- Modelling the LEON2 processor for SCoPE.
- Generation a SystemC file with the system description.

The AMT and AADS-T tools are developed as Eclipse plug-ins integrated with the OSATE AADL editor.

4 Case study

With the goal to exercise the HW/SW co-design methodology, a case study was developed based on a simplified space application for digital image processing. Figure 2 shows the different components of this case study.

The execution platform is composed of two different processing boards:

FPGA design board: it hosts the HW system that consists of the Image Processing System, formed by the Image Processing Module, Science Data Reporting Module and Control Module.

LEON2 design board: it hosts the SW system that consists of the On-Board SW (OBSW).

Moreover, the camera is replaced by a PC that communicates to the FPGA through a serial bus. The figure 3 shows the platform used in the HWSWCO case study. This case study exercises the whole methodology and rose important conclusions with regard to the interaction between the HW and SW synthesis phases.

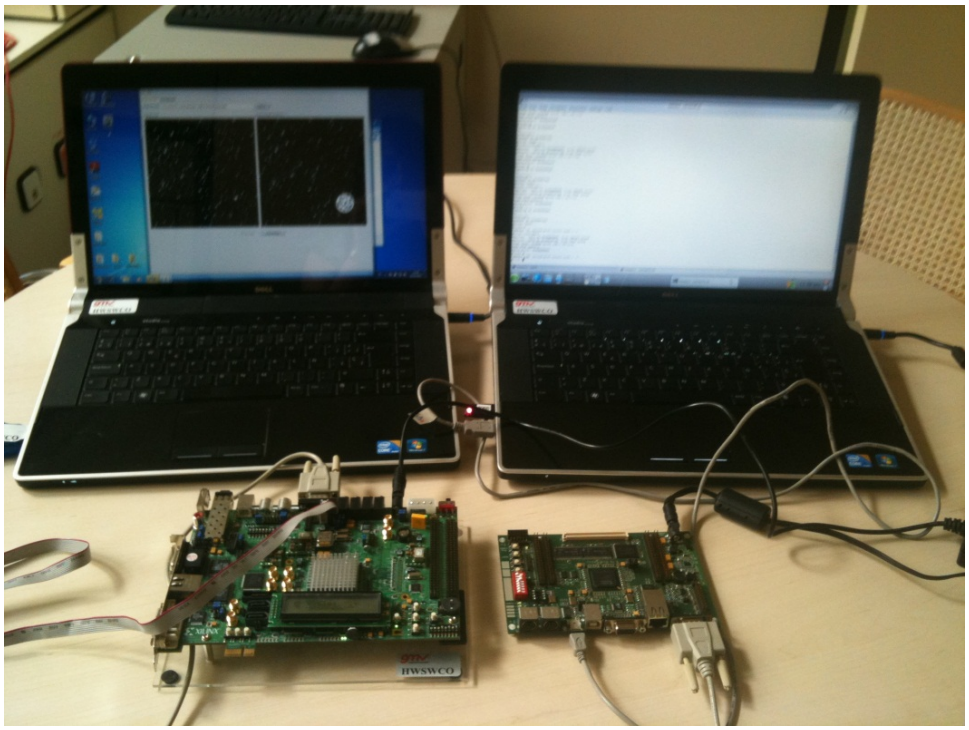

Fig. 3. HWSWCO demonstrator: FPGA Virtex 5, Leon2 development board and PC acting as camera instrument.

5 Conclusions and future work

Four key points were identified in the development of a distributed HW/SW system compliant with the ASSERT process:

- The modelling environment should be able to generate HW data model compatible with ASSERT, especially in terms of data size and bit ordering.
- The HW system must be compliant with the ASSERT component model.
- The HW system must have a common representation of the system components so that the ASSERT HW broker is able to dispatch the services mapped to HW, and communicate with the SW system using the same communication protocol.
- Finally, it should use a similar programming language so that the effort of migrating system functions from SW to HW is minimized to the maximum extent.

However this study did not cover all aspects of the HW/SW co-development, and there are some missing issues to be covered in future lines of work. One of the most interested is the analysis and definition of a computational model for the hardware systems that were analysable and compatible with the Ravenscar Computational model of the SW system.

Acknowledgments

This work was supported by the HWSWCO project ("Hardware/Software Co-design"), ESA Technological Research Program (TRP) study, contract ESTEC 22810/09/JK, directed by Dc. Eric Conquet. The partners involved in this study

were GMV as prime contractor, GMV Systems, the Universidad of Cantabria and the Universidad Politécnica of Madrid.

References

1. : ISO/IEC TR 24718:2005 — Guide for the use of the Ada Ravenscar Profile in high integrity systems. (2005) Based on the University of York Technical Report YCS-2003-348 (2003).
2. : ISO/IEC 8652:1995(E)/TC1(2000)/AMD1(2007): Information Technology — Programming Languages — Ada